Approximating the Diameter of Planar Graphs in Near Linear Time

Oren Weimann and Raphael Yuster

University of Haifa, Israel
{oren@cs.haifa.ac.il, raphy@math.haifa.ac.il}

Abstract. We present a $(1+\varepsilon)$-approximation algorithm running in $O(f(\varepsilon)\cdot n\log^4 n)$ time for finding the diameter of an undirected planar graph with non-negative edge lengths.

1 Introduction

The diameter of a graph is the largest distance between two vertices. Computing it is among the most fundamental algorithmic graph problems. In general weighted graphs, as well as in planar graphs, the only known way to compute the diameter is to essentially solve the (more general) All-Pairs Shortest Paths (APSP) problem and then take the pair of vertices with the largest distance.

In general weighted graphs, solving APSP (and thus diameter) currently requires $\tilde{O}(n^3)$ time. The fastest algorithm to date is $O(n^3(\log\log n)^3/\log^2 n)$ by Chan [5], or for sparse graphs $O(mn+n^2\log n)$ by Johnson [12], with a small improvement to $O(mn+n^2\log\log n)$ in [18].

In weighted *planar* graphs, solving APSP can be done in $O(n^2)$ time by Frederickson [10]. While this is optimal for APSP, it is not clear that it is optimal for diameter. Currently, only a logarithmic factor improvement by Wulf-Nilsen [20] is known for the diameter, running in $O(n^2(\log\log n)^4/\log n)$ time. A long standing open problem [6] is to find the diameter in truly subquadratic $O(n^{2-\varepsilon})$ time. Eppstein [8] has shown that if the diameter in a planar graph is bounded by a fixed constant then it can be found in $O(n)$ time. Fast algorithms are also known for some simpler classes of graphs like outer-planar graphs [9], interval graphs [17], and distance-hereditary graphs [7].

In lack of truly subcubic-time algorithms for general graphs and truly subquadratic-time algorithms for planar graphs it is natural to seek faster algorithms that *approximate* the diameter. It is easy to approximate the diameter within a factor of 2 by simply computing a Single-Source Shortest Path (SSSP) tree from any vertex in the graph and returning twice the depth of the deepest node in the tree. This requires $O(m+n\log n)$ time for general graphs and $O(n)$ time for planar graphs [11]. For general graphs, Aingworth et al. [2] improved the approximation factor from 2 to 3/2 at the cost of $\tilde{O}(m\sqrt{n}+n^2)$ running time, and Boitmanis et al. [4] gave an additive approximation factor of $O(\sqrt{n})$ with $\tilde{O}(m\sqrt{n})$ running time. For planar graphs, the current best approximation is a 3/2-approximation by Berman and Kasiviswanathan running in $O(n^{3/2})$ time [3]. We improve this to a $(1+\varepsilon)$-approximation running in $\tilde{O}(n)$ time for any fixed $0<\varepsilon<1$.

Our Result.

Theorem 1. *Given an undirected planar graph with non-negative edge lengths and diameter d, for any $\varepsilon>0$ we can compute an approximate diameter d' (where $d<d'<(1+\varepsilon)\cdot d$) in time $O(n\log^4 n/\varepsilon^4+n\cdot 2^{O(1/\varepsilon)})$.*

Summary of the Algorithm. A lemma of Lipton and Tarjan [15] states that, for any SSSP tree T in a planar graph, there is a non-tree edge e (where e might possibly be a non-edge of the planar graph) such that the strict interior and strict exterior of the unique simple cycle C in $T \cup \{e\}$ each contains at most $2/3 \cdot n$ vertices. The vertices of C therefore form a *separator* consisting of two shortest paths with the same common starting vertex.

Let G_{in} (resp. G_{out}) be the subgraph of G induced by C and all interior (resp. exterior) vertices to C. Let $d(G_{in}, G_{out}, G)$ denote the largest distance in the graph G between a *marked* vertex in $V(G_{in})$ and a *marked* vertex in $V(G_{out})$. In the beginning, all vertices of G are marked and we seek the diameter which is $d(G, G, G)$. We use a divide and conquer algorithm that first approximates $d(G_{in}, G_{out}, G)$, then unmarks all vertices of C, and then recursively approximates $d(G_{in}, G_{in}, G)$ and $d(G_{out}, G_{out}, G)$ and takes the maximum of all three. We outline this algorithm below. Before running it, we compute an SSSP tree from any vertex using the linear-time SSSP algorithm of Henzinger et al. [11]. The depth of the deepest node in this tree already gives a 2-approximation to the diameter $d(G, G, G)$. Let x be the obtained value such that $x \leq d(G, G, G) \leq 2x$.

Reduce $d(G_{in}, G_{out}, G)$ to $d(G_{in}, G_{out}, G_t)$ in a tripartite graph G_t: The separator C is composed of two shortest paths P and Q emanating from the same vertex, but that are otherwise disjoint. We carefully choose a subset of $16/\varepsilon$ vertices from C called *portals*. The first (resp. last) $8/\varepsilon$ portals are evenly spread vertices across the prefix of P (resp. Q) of length $8x$. The purpose of the portals is to approximate a shortest u-to-v path for $u \in G_{in}$ and $v \in G_{out}$ by forcing it to go through a portal. Formally, we construct a tripartite graph G_t with vertices $(V(G_{in}), \mathit{portals}, V(G_{out}))$. The length of edge $(u \in V(G_{in}), v \in \mathit{portals})$ or $(u \in \mathit{portals}, v \in V(G_{out}))$ in G_t is the u-to-v distance in G. This distance is computed by running the SSSP algorithm of [11] from each of the $16/\varepsilon$ portals. By the choice of portals, we show that $d(G_{in}, G_{out}, G_t)$ is a $(1 + 2\varepsilon)$-approximation of $d(G_{in}, G_{out}, G)$.

Approximate $d(G_{in}, G_{out}, G_t)$: If ℓ is the maximum edge-length of G_t, then note that $d(G_{in}, G_{out}, G_t)$ is between ℓ and 2ℓ. This fact makes it possible to round the edge-lengths of G_t to be in $\{1, 2, \ldots, 1/\varepsilon\}$ so that $\varepsilon\ell \cdot d(G_{in}, G_{out}, G_t)$ after rounding is a $(1 + 2\varepsilon)$-approximation to $d(G_{in}, G_{out}, G_t)$ before rounding. Using the fact that after rounding $d(G_{in}, G_{out}, G_t)$ is constant, we give a linear-time algorithm to compute it exactly, thus approximating $d(G_{in}, G_{out}, G)$. We then unmark all vertices of C and move on to recursively approximate $d(G_{in}, G_{in}, G)$ (the case of $d(G_{out}, G_{out}, G)$ is symmetric).

Reduce $d(G_{in}, G_{in}, G)$ to $d(G_{in}, G_{in}, G^+_{in})$ in a planar graph G^+_{in} of size at most $2/3 \cdot n$: In order to apply recursion, we construct a planar graph G^+_{in} that has at most $2/3 \cdot n$ vertices and $d(G_{in}, G_{in}, G^+_{in})$ is a $(1 + \varepsilon/(2 \log n))$-approximation[1] to $d(G_{in}, G_{in}, G)$. To construct G^+_{in}, we first carefully choose a subset of $256 \log n/\varepsilon$ vertices from C called *dense portals*. We then compute all $O((256 \log n/\varepsilon)^2)$ shortest paths in G_{out} between dense portals. The graph B' obtained by the union of all these paths has at most $O((256 \log n/\varepsilon)^4)$ vertices of degree > 2. We contract vertices of degree $= 2$ so that the number of vertices in B' decreases to $O((256 \log n/\varepsilon)^4)$. Appending this small graph B' (after unmarking all of its vertices) as an exterior to G_{in} results in a graph G^+_{in} that has $|G_{in}| + O((256 \log n/\varepsilon)^4)$ vertices and $d(G_{in}, G_{in}, G^+_{in})$ is a $(1 + \varepsilon/(2 \log n))$-approximation of $d(G_{in}, G_{in}, G)$.

The problem is still that the size of G^+_{in} is not necessarily bounded by $2/3 \cdot n$. This is because C (that is part of G^+_{in}) can be as large as n. We show how to shrink G^+_{in} to size roughly $2/3 \cdot n$ while

[1] $\log n = \log_2 n$ throughout the paper.

$d(G_{in}, G_{in}, G^+_{in})$ remains a $(1 + \varepsilon/(2\log n))$-approximation of $d(G_{in}, G_{in}, G)$. To achieve this, we shrink the C part of G^+_{in} so that it only includes the dense portals without changing $d(G_{in}, G_{in}, G^+_{in})$.

Approximate $d(G_{in}, G_{in}, G^+_{in})$*:* Finally, once $|G^+_{in}| \leq 2/3 \cdot n$ we apply recursion to $d(G_{in}, G_{in}, G^+_{in})$. In the halting condition, when $|G^+_{in}| \leq (256 \log n/\varepsilon)^4$, we naively compute $d(G_{in}, G_{in}, G^+_{in})$ using APSP.

Related Work. The use of shortest-path separators and portals to approximate distances in planar graphs was first suggested in the context of *approximate distance oracles*. These are data structures that upon query u, v return a $(1 + \varepsilon)$-approximation of the u-to-v distance. Thorup [19] presented an $O(1/\varepsilon \cdot n \log n)$-space oracle answering queries in $O(1/\varepsilon)$ time on directed weighted planar graphs. Independently, Klein [14] achieved these same bounds for undirected graphs.

In distance oracles, we need distances between every pair of vertices and each vertex is associated with a possibly different set of portals. In our diameter case however, since we know the diameter is between x and $2x$, it is possible to associate all vertices with the exact same set of portals. This fact is crucial in our algorithm, both for its running time and for its use of rounding. Another important distinction between our algorithm and distance oracles is that distance oracles upon query (u, v) can inspect all recursive subgraphs that include both u and v. We on the other hand must have that, for every (u, v), the shortest u-to-v path exists (approximately) in the unique subgraph where u and v are separated by C. This fact necessitated our construction of G^+_{in} and G^+_{out}.

2 The Algorithm

In this section we give a detailed description of an algorithm that approximates the diameter of an undirected weighted planar graph $\mathcal{G} = (V, E)$ in the bounds of Theorem 1. The algorithm computes a $(1 + \varepsilon)$-approximation of the diameter $d = d(G, G, G)$ for $G = \mathcal{G}$. This means it returns a value d' where $d \leq d' \leq (1 + \varepsilon) \cdot d$ (recall that, before running the algorithm, we compute a value x such that $x \leq d \leq 2x$ by computing a single-source shortest-path tree from an arbitrary vertex in $\mathcal{G}$). We focus on approximating the value of the diameter. An actual path of length d' can be found in the same time bounds. For simplicity we will assume that shortest paths are unique. This can always be achieved by adding random innitesimal weights to each edge; using the isolation lemma of [16]. Also, to simplify the presentation, we assume that $\varepsilon \leq 0.1$ and we describe a $(1+7\varepsilon)$-approximation. (then just take $\varepsilon' = \varepsilon/7$).

The algorithm is recursive and actually solves the more general problem of finding the largest distance only between all pairs of *marked vertices*. In the beginning, we mark all $n = |V(G)|$ vertices of $G = \mathcal{G}$ and approximate $d(G, G, G)$ (the largest distance in G between marked vertices in $V(G)$). Throughout the recursive calls, we maintain the invariant that the distance between any two *marked* vertices in the graph G of the recursive call is a $(1+\varepsilon)$-approximation of their distance in the original graph $\mathcal{G}$ (there is no guarantee on the marked-to-unmarked or the unmarked-to-unmarked distances). We denote by $\delta_{\mathcal{G}}(u, v)$ the u-to-v distance in the original graph $\mathcal{G}$.

The recursion is applied according to a variant of the shortest-path separator decomposition for planar graphs by Lipton and Tarjan [15]: We first pick any *marked* vertex v_1 and compute in linear time the SSSP tree from v_1 in G. In this tree, we can find in linear time two shortest paths P and Q (both emanating from v_1) such that removing the vertices of $C = P \cup Q$ from G results in two disjoint planar subgraphs A and B (i.e., there are no edges in $V(A) \times V(B)$). The number of vertices of C can be as large as n but it is guaranteed that $|V(A)| \leq 2/3 \cdot n$ and $|V(B)| \leq 2/3 \cdot n$.

Notice that the paths P and Q might share a common prefix. It is common to not include this shared prefix in C. However, in our case, we must have the property that P and Q start at a *marked* vertex. So we include in C the shared prefix as well (See Fig. 1).

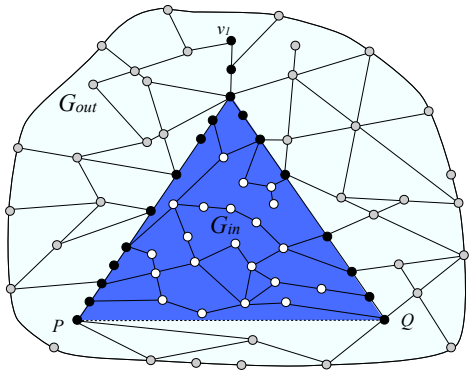

Fig. 1. Shortest-path separator: In the above weighted undirected planar graph G, the black nodes constitute the shortest path separator C composed of two shortest paths P and Q emanating from the same vertex v_1. The subgraph of G induced by the white nodes is denoted A and the subgraph of G induced by the gray nodes is denoted B. The graph G_{in} is the subgraph induced by all white and black vertices and the graph G_{out} is the subgraph induced by all gray and black vertices.

Let G_{in} (resp. G_{out}) be the subgraph of G induced by $V(C) \cup V(A)$ (resp. $V(C) \cup V(B)$). To approximate $d(G, G, G)$, we first compute a $(1 + 5\varepsilon)$-approximation d_1 of $d(G_{in}, G_{out}, G)$ (the largest distance in G between the marked vertices of $V(G_{in})$ and the marked vertices of $V(G_{out})$). In particular, d_1 takes into account all $V(C) \times V(G)$ distances. We can therefore unmark all the vertices of C and move on to approximate $d_2 = d(G_{in}, G_{in}, G)$ (approximating $d_3 = d(G_{out}, G_{out}, G)$ is done similarly). We approximate $d(G_{in}, G_{in}, G)$ by applying recursion on $d(G_{in}, G_{in}, G^+_{in})$ where $|V(G^+_{in})| \leq 2/3 \cdot n$. The marked vertices in G^+_{in} and in G_{in} are the same and $d(G_{in}, G_{in}, G^+_{in})$ is a $(1 + \varepsilon/(2 \log n))$-approximation of $d(G_{in}, G_{in}, G)$. This way, we add a factor of $\varepsilon/(2 \log n)$ to the diameter in each recursive call. Since the recursive depth is $O(\log n)$ (actually, it is never more than $1.8 \log n$) we get a $(1+5\varepsilon) \cdot (1+\varepsilon) \leq (1+7\varepsilon)$-approximation d_2 to $d(G_{in}, G_{in}, G)$. Finally, we return the maximum of d_1, d_2, d_3.

2.1 Reduce $d(G_{in}, G_{out}, G)$ to $d(G_{in}, G_{out}, G_t)$

Our goal is now to approximate $d(G_{in}, G_{out}, G)$. For $u \in G_{in}$ and $v \in G_{out}$, we approximate a shortest u-to-v path in G by forcing it to go through a *portal*. In other words, consider a shortest u-to-v path. It is is obviously composed of a shortest u-to-c path in G concatenated with a shortest c-to-v path in G for some vertex $c \in C$. We approximate the shortest u-to-v path by insisting that c is a portal. The fact that we only need to consider u-to-v paths that are of length between x and $2x$ makes it possible to choose the same portals for all vertices.

We now describe how to choose the portals in linear time. Recall that the separator C is composed of two shortest paths P and Q emanating from the same *marked* vertex v_1. The vertex v_1 is chosen as the first portal. Then, for $i = 2, \ldots, 8/\varepsilon$ we start from v_{i-1} and walk on P until we

reach the first vertex whose distance from v_{i-1} via P is greater than εx. We set this vertex as the portal v_i and continue to $i+1$. Notice that P is a shortest path in G and the portals are chosen in a prefix of P of length $8x$. This might seem counterintuitive as we know that any shortest path P in the original graph $\mathcal{G}$ is of length at most $2x$. However, since one endpoint of P is not necessarily marked, it is possible that P is a shortest path in G but not even an approximate shortest path in the original graph $\mathcal{G}$. We do the same for Q, and we get a total of $16/\varepsilon$ portals (See Fig. 2).

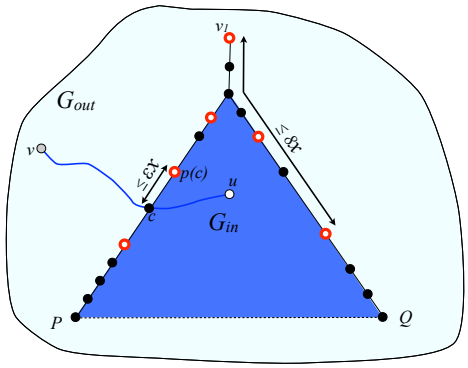

Fig. 2. Portals: The six circled vertices are the $16/\varepsilon$ portals in the $8x$ prefixes of P and Q. The shortest path between u and v goes through the separator vertex c and is approximated by the u-to-$p(c)$ and the $p(c)$-to-v shortest paths where $p(c)$ is the closest portal to c. The distance from c to $p(c)$ is at most εx.

Once we have chosen the portals, we move on to construct a tripartite graph G_t whose three vertex sets are $(V(G_{in}), portals, V(G_{out}))$. The length of edge $(u \in V(G_{in}), v \in portals)$ or $(u \in portals, v \in V(G_{out}))$ is the u-to-v distance in G. This distance is computed by running the linear-time SSSP algorithm of Henzinger et al. [11] in G from each of the $16/\varepsilon$ portals in total $O(1/\varepsilon \cdot |V(G)|)$ time. The following lemma states that our choice of portals implies that $d(G_{in}, G_{out}, G_t)$ is a good approximation of $d(G_{in}, G_{out}, G)$.

Lemma 1. *If $d(G_{in}, G_{out}, G) \geq x$, then $d(G_{in}, G_{out}, G_t)$ is a $(1+2\varepsilon)$-approximation of $d(G_{in}, G_{out}, G)$. If $d(G_{in}, G_{out}, G) < x$ then $d(G_{in}, G_{out}, G_t) \leq (1+2\varepsilon) \cdot x$.*

Proof. The first thing to notice is that $d(G_{in}, G_{out}, G_t) \geq d(G_{in}, G_{out}, G)$. This is because every shortest u-to-v path in G_t between a marked vertex $u \in V(G_{in})$ of the first column and a marked vertex $v \in V(G_{out})$ of the third column corresponds to an actual u-to-v path in G.

We now show that $d(G_{in}, G_{out}, G_t) \leq (1+2\varepsilon) \cdot d(G_{in}, G_{out}, G)$. We begin with some notations. Let P_t denote the shortest path in G_t realizing $d(G_{in}, G_{out}, G_t)$. The path P_t is a shortest u-to-v path for some *marked* vertices $u \in G_{in}$ and $v \in G_{out}$. The length of the path P_t is $\delta_{G_t}(u, v)$. Let P_G denote the shortest u-to-v path in G that is of length $\delta_G(u, v)$ and let $P_{\mathcal{G}}$ denote the shortest u-to-v path in the original graph $\mathcal{G}$ that is of length $\delta_{\mathcal{G}}(u, v)$. Recall that we have the invariant that in every recursive level for every pair of marked vertices $\delta_G(u, v) \leq (1+\varepsilon) \cdot \delta_{\mathcal{G}}(u, v)$. We also have that $\delta_{\mathcal{G}}(u, v) \leq 2x$ and so $\delta_G(u, v) \leq 2x \cdot (1+\varepsilon)$. For the same reason, since v_1 (the first vertex of both P and Q) is also marked, we know that $\delta_G(v_1, u)$ is of length at most $2x \cdot (1+\varepsilon)$.

The path P_G must include at least one vertex $c \in C$. Assume without loss of generality that $c \in P$. We claim that c must be a vertex in the prefix of P of length $8x$. Assume the converse, then the v_1-to-c prefix of P is of length at least $8x$. Since P is a shortest path in G, this means that $\delta_G(v_1, c)$ is at least $8x$. However, consider the v_1-to-c path composed of the v_1-to-u shortest path (of length $\delta_G(v_1, u) \leq 2x \cdot (1+\varepsilon)$) concatenated with the u-to-c shortest path (of length $\delta_G(u, c) \leq \delta_G(u, v) \leq 2x \cdot (1+\varepsilon)$). Their total length is $4x \cdot (1+\varepsilon)$ which is less than $8x$ (since $\varepsilon < 1$) thus contradicting our assumption.

After establishing that c is somewhere in the $8x$ prefix of P, we now want to show that $\delta_{G_t}(u, v) \leq (1+2\varepsilon) \cdot \delta_G(u, v)$. Let $p(c)$ denote the closest portal to c on the path P. Notice that by our choice of portals and since c is in the $8x$ prefix of P we have that $\delta_G(c, p(c)) \leq \varepsilon x$. By the triangle inequality we know that $\delta_G(u, p(c)) \leq \delta_G(u, c) + \delta_G(c, p(c)) \leq \delta_G(u, c) + \varepsilon x$ and similarly $\delta_G(p(c), v) \leq \delta_G(c, v) + \varepsilon x$. This means that

$$\begin{aligned} d(G_{in}, G_{out}, G_t) &= \delta_{G_t}(u, v) \\ &\leq \delta_G(u, p(c)) + \delta_G(p(c), v) \\ &\leq \delta_G(u, c) + \delta_G(c, v) + 2\varepsilon x \\ &= \delta_G(u, v) + 2\varepsilon x \\ &\leq d(G_{in}, G_{out}, G) + 2\varepsilon x \\ &\leq (1+2\varepsilon) \cdot d(G_{in}, G_{out}, G). \end{aligned}$$

In the last inequality we assumed that $d(G_{in}, G_{out}, G) \geq x$. If $d(G_{in}, G_{out}, G) < x$, then we get that $d(G_{in}, G_{out}, G_t) \leq (1+2\varepsilon) \cdot x$. The lemma follows. □

By Lemma 1, approximating $d(G_{in}, G_{out}, G)$ when $d(G_{in}, G_{out}, G) \geq x$ reduces to approximating $d(G_{in}, G_{out}, G_t)$. The case of $d(G_{in}, G_{out}, G) < x$ means that the diameter d of the original graph $\mathcal{G}$ is *not* a $(u \in G_{in})$-to-$(v \in G_{out})$ path. This is because $d \geq x > d(G_{in}, G_{out}, G) \geq d(G_{in}, G_{out}, \mathcal{G})$. So d will be approximated in a different recursive call (when the separator separates the endpoints of the diameter). In the meanwhile, we will get that $d(G_{in}, G_{out}, G_t)$ is at most $(1+2\varepsilon) \cdot x$ and so it will not compete with the correct recursive call when taking the maximum.

2.2 Approximate $d(G_{in}, G_{out}, G_t)$

In this subsection, we show how to approximate the diameter in the tripartite graph G_t. We give a $(1+2\varepsilon)$-approximation for $d(G_{in}, G_{out}, G_t)$. By the previous subsection, this means we have a $(1+2\varepsilon)(1+2\varepsilon) < (1+5\varepsilon)$-approximation for $d(G_{in}, G_{out}, G)$. From the invariant that distances in G between marked vertices are a $(1+\varepsilon)$-approximation of these distances in the original graph $\mathcal{G}$, we get a $(1+5\varepsilon)(1+\varepsilon) < (1+7\varepsilon)$-approximation for $d(G_{in}, G_{out}, \mathcal{G})$ in the original graph $\mathcal{G}$.

We now present our $(1+2\varepsilon)$-approximation for $d(G_{in}, G_{out}, G_t)$ in the tripartite graph G_t. Recall that P_t denotes the shortest path in G_t realizing $d(G_{in}, G_{out}, G_t)$. By the definition of G_t, we know that the path P_t is composed of only two edges: (1) edge (u, p) between a marked vertex u of the first column (i.e., $u \in V(G_{in})$) and a vertex p of the second column (i.e., p corresponds to some portal in G). (2) edge (p, v) between p and a marked vertex v of the third column (i.e., $v \in V(G_{out})$).

Let X (resp. Y) denote the set of all edges in G_t adjacent to marked vertices of the first (resp. third) column. Let ℓ denote the maximum edge-length over all edges in $X \cup Y$. Notice that

$\ell \leq d(G_{in}, G_{out}, G_t) \leq 2\ell$. We round up the lengths of all edges in $X \cup Y$ to the closest multiple of $\varepsilon\ell$. The rounded edge-lengths are thus all in $\{\varepsilon\ell, 2\varepsilon\ell, 3\varepsilon\ell, \ldots, \ell\}$. We denote G_t after rounding as G'_t. Notice that $d(G_{in}, G_{out}, G'_t)$ is a $(1+2\varepsilon)$-approximation of $d(G_{in}, G_{out}, G_t)$. This is because the path P_t is of length at least ℓ and is composed of two edges, each one of them has increased its length by at most $\varepsilon\ell$.

We now show how to compute $d(G_{in}, G_{out}, G'_t)$ exactly in linear time. We first divide all the edge-lengths of G'_t by $\varepsilon\ell$ and get that G'_t has edge-lengths in $\{1, 2, 3, \ldots, 1/\varepsilon\}$. After finding $d(G_{in}, G_{out}, G'_t)$ (which is now a constant) we simply multiply the result by $\varepsilon\ell$. The following lemma states that when the diameter is constant it is possible to compute it exactly in linear time. Note that we can't just use Eppstein's [8] linear-time diameter algorithm for a planar graph whose diameter is bounded by a fixed constant since in our case we get a *non-planar* tripartite graph G'_t.

Lemma 2. *$d(G_{in}, G_{out}, G'_t)$ can be computed exactly in time $O(|V(G)|/\varepsilon + 2^{O(1/\varepsilon)})$.*

Proof. Recall that in G'_t we denote the set of all edges adjacent to marked vertices of the first and third column as X and Y. The length of each edge in $X \cup Y$ is in $\{1, 2, \ldots, 1/\varepsilon\}$. The number of edges in X (and similarly in Y) is at most $16|V(G)|/\varepsilon$. This is because the first column contains $|G_{in}| \leq |V(G)|$ vertices and the second column contains $16/\varepsilon$ vertices $v_1, v_2, \ldots, v_{16/\varepsilon}$ (the portals).

For every marked vertex v in the first (reps. third) column, we store a $16/\varepsilon$-tuple v_X (reps. v_Y) containing the edge lengths from v to all vertices of the second column. In other words, the tuple $v_X = \langle \delta(v, v_1), \delta(v, v_2), \ldots, \delta(v, v_{16/\varepsilon}) \rangle$ where $\delta(v, v_i)$ is the length of the edge (v, v_i).

Notice that every $\delta(v, v_i)$ is in $\{1, 2, \ldots, 1/\varepsilon\}$. Furthermore, every $|\delta(v, v_{i+1}) - \delta(v, v_i)|$ is in $\{0, 1, 2\}$. To see this, notice that v_i and v_{i+1} are adjacent portals on the separator (P or Q). So the distance between them in G is $\delta(v_i, v_{i+1}) \leq \varepsilon x$, and since we scaled (divided by $\varepsilon\ell$) $\delta(v_i, v_{i+1})$ is at most $\varepsilon x/\varepsilon\ell$. Because $x \leq 2\ell$ we get that $\varepsilon x/\varepsilon\ell \leq 2$. Finally, since $\delta(v, v_i)$ and $\delta(v, v_{i+1})$ correspond to distances, by the triangle inequality we have $|\delta(v, v_{i+1}) - \delta(v, v_i)| \leq \delta(v_i, v_{i+1}) \leq 2$. Overall, we get that a tuple v_X or v_Y has $16/\varepsilon$ entries, the first entry is $\delta(v, v_1) \in \{1, 2, \ldots, 1/\varepsilon\}$ and for every other entry $i+1$ it holds that $\delta(v, v_{i+1}) - \delta(v, v_i)$ can be one of 5 values: $-2, -1, 0, 1, 2$. The total number of tuples is $16|V(G)|/\varepsilon$ but the number of *different* tuples is therefore only $k = (1/\varepsilon) \cdot 5^{16/\varepsilon}$.

We create two binary vectors V_X and V_Y each of length k. The i'th bit of V_X (reps. V_Y) is 1 iff the i'th possible tuple exists as some v_X (reps. v_Y). Creating these vectors takes $O(|V(G)|/\varepsilon)$ time. Then, for every 1 bit in V_X (corresponding to a tuple of vertex u in the first column) and every 1 bit in V_Y (corresponding to a tuple of vertex v in the third column) we compute the u-to-v distance in G'_t using the two tuples in time $16/\varepsilon$. We then return the maximum of all such (u, v) pairs. Notice that a 1 bit can correspond to several vertices that have the exact same tuple. We arbitrarily choose any one of these. There are k entries in V_X and k entries in V_Y so there are $O(k^2)$ pairs of 1 bits. Each pair is examined in $O(16/\varepsilon)$ time for a total of $O((1/\varepsilon) \cdot k^2) = 2^{O(1/\varepsilon)}$ time. □

To conclude, we have so far seen how to obtain a $(1+5\varepsilon)$-approximation for $d(G_{in}, G_{out}, G)$ implying a $(1+7\varepsilon)$-approximation for $d(G_{in}, G_{out}, \mathcal{G})$ in the original graph $\mathcal{G}$. The next step is to unmark all vertices of C and move on to recursively approximate $d(G_{in}, G_{in}, G)$ (approximating $d(G_{out}, G_{out}, G)$ is done similarly).

2.3 Reduce $d(G_{in}, G_{in}, G)$ to $d(G_{in}, G_{in}, G^+_{in})$

In this subsection we show how to recursively obtain a $(1+5\varepsilon)$-approximation of $d(G_{in}, G_{in}, G)$ and recall that this implies a $(1+7\varepsilon)$-approximation of $d(G_{in}, G_{in}, \mathcal{G})$ in the original graph $\mathcal{G}$ since

we will make sure to maintain our invariant that, at any point of the recursion, distances between marked vertices are a $(1+\varepsilon)$-approximation of these distances in the original graph $\mathcal{G}$.

It is important to note that our desired construction can be obtained with similar guarantees using the construction of Thorup [19] for distance oracles. However, we present here a simpler construction than [19] since, as apposed to distance oracles that require *all-pairs* distances, we can afford to only consider distances that are between x and $2x$.

There are two problems with applying recursion in order to solve $d(G_{in}, G_{in}, G)$. The first is that $|V(G_{in})|$ can be as large as $|V(G)|$ and we need it to be at most $2/3 \cdot |V(G)|$. We do know however that the number of *marked* vertices in $V(G_{in})$ is at most $2/3 \cdot |V(G)|$. The second problem is that it is possible that the u-to-v shortest path in G for $u, v \in G_{in}$ includes vertices of G_{out}. This only happens if the u-to-v shortest path in G is composed of a shortest u-to-p path ($p \in P$) in G_{in}, a shortest p-to-q path ($q \in Q$) in G_{out}, and a shortest q-to-v path in G_{in}. To overcome these two problems, we construct a planar graph G^+_{in} that has at most $2/3 \cdot |V(G)|$ vertices and $d(G_{in}, G_{in}, G^+_{in})$ is a $(1+\varepsilon/(2\log n))$-approximation to $d(G_{in}, G_{in}, G)$.

Recall that the subgraph B of G induced by all vertices in the strict exterior of the separator C is such that $|B| \leq 2/3 \cdot |V(G)|$ and $G_{out} = B \cup C$. The construction of G^+_{in} is done in two phases. In the first phase, we replace the B part of G with a graph B' of polylogarithmic size. In the second phase, we contract the C part of G to polylogarithmic size.

Phase I: replacing B with B'. To construct G^+_{in}, we first choose a subset of $256 \log n/\varepsilon$ vertices from C called *dense portals*. The dense portals are chosen similarly to the regular portals but there are more of them. The marked vertex v_1 (the first vertex of both P and Q) is chosen as the first dense portal. Then, for $i = 2, \ldots, 128 \log n/\varepsilon$ we start from v_{i-1} and walk on P until we reach the first vertex whose distance from v_{i-1} via P is greater than $\varepsilon x/(16 \log n)$. We set this vertex as the dense portal v_i and continue to $i+1$. We do the same for Q, for a total of $256 \log n/\varepsilon$ dense portals.

After choosing the dense portals, we compute all $O((256 \log n/\varepsilon)^2)$ shortest paths in G_{out} between dense portals. This can be done using SSSP from each portal in total $O(|V(G_{out})| \cdot \log n/\varepsilon)$ time. It can also be done using the Multiple Source Shortest Paths (MSSP) algorithm of Klein [13] in total $O(|V(G_{out})| \cdot \log n + \log^2 n/\varepsilon^2)$ time.

Let B' denote the graph obtained by the union of all these dense portal to dense portal paths in G_{out}. Notice that since these are shortest paths, and since we assumed shortest paths are unique, then every two paths can share at most one consecutive subpath. The endpoints of this subpath are of degree > 2. There are only $O((256 \log n/\varepsilon)^2)$ paths so this implies that the graph B' has at most $O((256 \log n/\varepsilon)^4)$ vertices of degree > 2. We can therefore contract vertices of degree $= 2$. The number of vertices of B' then decreases to $O((256 \log n/\varepsilon)^4)$, it remains a planar graph, and its edge lengths correspond to subpath lengths.

We then unmark all vertices of B' and append B' to the infinite face of G_{in}. In other words, we take the disjoint union of G_{in} and B' and identify the dense portals of G_{in} with the dense portals of B'. This results in a graph G^+_{in} that has $|V(G_{in})| + O((256 \log n/\varepsilon)^4)$ vertices. In Lemma 3 we will show that $d(G_{in}, G_{in}, G^+_{in})$ can serve as a $(1+\varepsilon/(2\log n))$-approximation to $d(G_{in}, G_{in}, G)$. But first we will shrink G^+_{in} so that the number of its vertices is bounded by $2/3 \cdot |V(G)|$.

Phase II: shrinking G^+_{in}. The problem with the current G^+_{in} is still that the size of $V(G^+_{in})$ is not necessarily bounded by $2/3 \cdot |V(G)|$. This is because C (that is part of $V(G^+_{in})$) can be as large as n. We now show how to shrink $V(G^+_{in})$ to size $2/3 \cdot |V(G)|$ while $d(G_{in}, G_{in}, G^+_{in})$ remains a

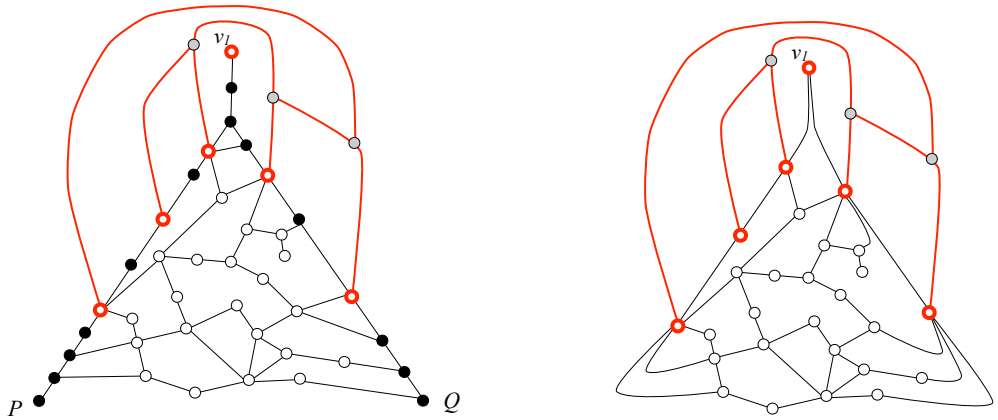

Fig. 3. On the left: The graph G^+_{in} before shrinking. The white vertices are the vertices of A, the black vertices are the vertices of C that are not dense portals, the six red circled vertices are the $256 \log n/\varepsilon$ dense portals, and the gray vertices are the vertices of $B' \setminus C$ with degree > 2. On the right: The graph G^+_{in} after shrinking. The edges adjacent to vertices of C that are not dense portals are now replaced with edges to dense portals.

$(1 + \varepsilon/(2 \log n))$-approximation of $d(G_{in}, G_{in}, G)$. To achieve this, we shrink the C part of $V(G^+_{in})$ so that it only includes the dense portals. We show how to shrink P, shrinking Q is done similarly.

Consider two dense portals v_i and v_{i+1} on P (i.e., v_i is the closest portal to v_{i+1} on the path P towards v_1). We want to eliminate all vertices of P between v_i and v_{i+1}. Denote these vertices by $p_1, \ldots, p_k$. If v_i is the last portal of P (i.e., $i = 128 \log n$) then $p_1, \ldots, p_k$ are all the vertices between v_i and the end of P. Recall that A is the subgraph of G induced by all vertices in the strict interior of the separator C. Fix a planar embedding of G^+_{in}. We perform the following process as long as there is some vertex u in $Q \cup A$ which is a neighbor of some p_j, and which is on some face of the embedding that also contains v_i. We want to "force" any shortest path that goes through an edge (u, p_j) to also go through the dense portal v_i. To this end, we delete all such edges (u, p_j), and instead insert a single edge (u, v_i) of length $\min_j\{\ell(u, p_j) + \delta_G(p_j, v_i)\}$. Here, $\ell(u, p_j)$ denotes the length of the edge (u, p_j) (it may be that $\ell(u, p_j) = \infty$ if (u, p_j) is not an edge) and $\delta_G(p_j, v_i)$ denotes the length of the p_j-to-v_i subpath of P. It is important to observe that the new edge (u, v_i) can be embedded while maintaining the planarity since we have chosen u to be on the same face as v_i. Observe that once the process ends, the vertices p_j have no neighbors in $Q \cup A$.

Finally, we replace the entire v_{i+1}-to-v_i subpath of P with a single edge (v_{i+1}, v_i) whose length is equal to the entire subpath length. If v_i is the last dense portal in P then we simply delete the entire subpath between v_i and the end of P. The entire shrinking process takes only linear time in the size of $|V(G)|$ since it is linear in the number of edges of G^+_{in} (which is a planar graph).

The following Lemma asserts that after the shrinking phase $d(G_{in}, G_{in}, G^+_{in})$ can serve as a $(1 + \varepsilon/(2 \log n))$-approximation to $d(G_{in}, G_{in}, G)$.

Lemma 3. $d(G_{in}, G_{in}, G) \leq d(G_{in}, G_{in}, G^+_{in}) \leq d(G_{in}, G_{in}, G) + \varepsilon x/(2 \log n)$

Proof. First observe that $d(G_{in}, G_{in}, G^+_{in}) \geq d(G_{in}, G_{in}, G)$. This is because every vertex of G_{in} that is marked in G is also a marked vertex in G^+_{in}, and any shortest u-to-v path in G^+_{in} corresponds to an actual u-to-v path in G.

We now show that $d(G_{in}, G_{in}, G^+_{in}) \leq d(G_{in}, G_{in}, G) + \varepsilon x/(2\log n)$. Let P^+ denote the shortest u-to-v path in G^+_{in} realizing $d(G_{in}, G_{in}, G^+_{in})$. Both u and v are marked vertices in G_{in} and the length of P^+ is $\delta_{G^+_{in}}(u, v)$. Let P_G denote the shortest u-to-v path in G that is of length $\delta_G(u, v)$.

Case 1: If P_G does not include any vertex of C then P_G is also present in G^+_{in} and therefore $d(G_{in}, G_{in}, G^+_{in}) \leq d(G_{in}, G_{in}, G)$.

Case 2: If P_G includes vertices that are not in G_{in} (i.e., vertices in $G_{out} \setminus C$) then P_G must be composed of a shortest u-to-p path ($p \in P$) in G_{in}, a shortest p-to-q path ($q \in Q$) in G_{out}, and a shortest q-to-v path in G_{in}.

We first claim that p must be a vertex in the prefix of P of length $8x$ (a similar argument holds for q and Q). Assume the converse, then the prefix of P from v_1 (the first vertex of both P and Q) to p is of length at least $8x$. Recall that we have the invariant that in every recursive level for every pair of marked vertices $\delta_G(u, v) \leq (1+\varepsilon) \cdot \delta_{\mathcal{G}}(u, v) \leq 2x \cdot (1+\varepsilon)$. For the same reason we know that $\delta_G(v_1, u) \leq 2x \cdot (1+\varepsilon)$. Since P is a shortest path in G, this means that $\delta_G(v_1, p) \geq 8x$. However, consider the v_1-to-p path composed of the v_1-to-u shortest path (of length $\delta_G(v_1, u) \leq 2x \cdot (1+\varepsilon)$) concatenated with the u-to-p shortest path (of length $\delta_G(u, p) \leq \delta_G(u, v) \leq 2x \cdot (1+\varepsilon)$). Their total length is $4x \cdot (1+\varepsilon)$ which is less than $8x$ (since $\varepsilon < 1$) thus contradicting our assumption.

We now show that $\delta_{G^+_{in}}(u, v) \leq \delta_G(u, v) + \varepsilon x/(2\log n)$. For $c \in P$ (resp. $c \in Q$), let $p(c)$ denote the first dense portal encountered while walking from c towards v_1 on the path P (resp. Q). Notice that since p and q are in the $8x$ prefixes of P and Q we have that $\delta_G(p, p(p)) \leq \varepsilon x/(16\log n)$ and $\delta_G(q, p(q)) \leq \varepsilon x/(16\log n)$. From the shrinking phase, it is easy to see that G^+_{in} includes a u-to-$p(p)$ path of length $\delta_G(u, p) + \delta_G(p, p(p))$ and so $\delta_{G^+_{in}}(u, p(p)) \leq \delta_G(u, p) + \varepsilon x/(16\log n)$. Similarly, $\delta_{G^+_{in}}(p(q), v) \leq \delta_G(q, v) + \varepsilon x/(16\log n)$. Furthermore, since G^+_{in} was appended with shortest paths between dense portals in G_{out} we have $\delta_{G^+_{in}}(p(p), p(q)) \leq \delta_{G_{out}}(p(p), p) + \delta_{G_{out}}(p, q) + \delta_{G_{out}}(q, p(q)) = \delta_G(p(p), p) + \delta_{G_{out}}(p, q) + \delta_G(q, p(q)) \leq \delta_{G_{out}}(p, q) + \varepsilon x/(8\log n)$. To conclude we get that

$$\begin{aligned}
d(G_{in}, G_{in}, G^+_{in}) &= \delta_{G^+_{in}}(u, v) \\
&\leq \delta_{G^+_{in}}(u, p(p)) + \delta_{G^+_{in}}(p(p), p(q)) + \delta_{G^+_{in}}(p(q), v) \\
&\leq \delta_G(u, p) + \delta_{G_{out}}(p, q) + \delta_G(q, v) + \varepsilon x/(4\log n) \\
&= \delta_G(u, v) + \varepsilon x/(4\log n) \\
&\leq d(G_{in}, G_{in}, G) + \varepsilon x/(4\log n) \\
&< d(G_{in}, G_{in}, G) + \varepsilon x/(2\log n).
\end{aligned}$$

Case 3: Finally, we need to consider the case where P_G includes only vertices of G_{in}. We assume P_G includes vertices of P and\or vertices of Q (otherwise this was handled in Case 1). We focus on the case that P_G includes vertices of both P and Q. The case that P_G includes vertices of one of P or Q follows immediately using a similar argument.

Since P and Q are shortest paths, then P_G must be composed of the following shortest paths: a u-to-p path ($p \in P$) in G_{in}, a p-to-p' subpath ($p' \in P$) of P, a p'-to-q' path ($q' \in Q$) in G_{in}, a q'-to-q subpath ($q \in Q$) of Q, and a q-to-v path in G_{in}. Following the same argument as in Case 2, we know that p and p' (reps. q and q') must in the prefix of P (reps. Q) of length $8x$. This means $\delta_G(c, p(c)) \leq \varepsilon x/(16\log n)$ for every $c \in \{p, p', q, q'\}$.

From the shrinking phase, it is easy to see that G^+_{in} includes a u-to-$p(p)$ path of length $\delta_G(u, p) + \delta_G(p, p(p))$ and so $\delta_{G^+_{in}}(u, p(p)) \leq \delta_G(u, p) + \varepsilon x/(16\log n)$. Similarly, $\delta_{G^+_{in}}(p(q), v) \leq$

$\delta_G(q,v)+\varepsilon x/(16\log n)$, and $\delta_{G^+_{in}}(p(p'),p(q')) \le \delta_G(p(p'),p')+\delta_G(p',q')+\delta_G(q',p(q')) \le \delta_G(p',q')+\varepsilon x/(8\log n)$. Furthermore, since subpaths of P in G^+_{in} between dense portals capture their exact distance in G we have that $\delta_{G^+_{in}}(p(p),p(p')) \le \delta_G(p(p),p)+\delta_G(p,p')+\delta_G(p',p(p')) \le \delta_G(p,p')+\varepsilon x/(8\log n)$ and similarly $\delta_{G^+_{in}}(p(q'),p(q)) \le \delta_G(q',q)+\varepsilon x/(8\log n)$. To conclude we get that

$$\begin{aligned}
d(G_{in},G_{in},G^+_{in}) &= \delta_{G^+_{in}}(u,v)\\
&\le \delta_{G^+_{in}}(u,p(p))+\delta_{G^+_{in}}(p(p),p(p'))+\delta_{G^+_{in}}(p(p'),p(q'))+\delta_{G^+_{in}}(p(q'),p(q))+\delta_{G^+_{in}}(p(q),v)\\
&\le \delta_G(u,p)+\delta_G(p,p')+\delta_G(p',q')+\delta_G(q',q)+\delta_G(q,v)+\varepsilon x/(2\log n)\\
&= \delta_G(u,v)+\varepsilon x/(2\log n)\\
&< d(G_{in},G_{in},G)+\varepsilon x/(2\log n).
\end{aligned}$$

□

Corollary 1. *If $d(G_{in},G_{in},G) \ge x$ then $d(G_{in},G_{in},G^+_{in})$ is a $(1+\varepsilon/(2\log n))$-approximation of $d(G_{in},G_{in},G)$. If $d(G_{in},G_{in},G) < x$, then we get that $d(G_{in},G_{in},G^+_{in}) \le (1+\varepsilon/(2\log n))\cdot x$.*

By the above corollary, approximating $d(G_{in},G_{in},G)$ when $d(G_{in},G_{in},G) \ge x$ reduces to approximating $d(G_{in},G_{in},G^+_{in})$. When $d(G_{in},G_{in},G) < x$ it means that the diameter of the original graph $\mathcal{G}$ is *not* a $(u \in G_{in})$-to-$(v \in G_{in})$ path and will thus be approximated in a different recursive call.

Finally, notice that indeed we maintain the invariant that the distance between any two *marked* vertices in the recursive call to G^+_{in} is a $(1+\varepsilon)$-approximation of the distance in the original graph $\mathcal{G}$. This is because, by the above corollary, every recursive call adds a $1+\varepsilon/(2\log n)$ factor to the approximation. Each recursive call decreases the input size by a factor of $(2/3+o(1))^{-1}$. Hence, the overall depth of the recursion is at most $\log_{1.5-o(1)} n < 1.8\log n$. Since

$$(1+\varepsilon/(2\log n))^{1.8\log n} < e^{0.9\varepsilon} < 1+\varepsilon$$

the invariant follows (we assume in the last inequality that $\varepsilon \le 0.1$). Together with the $(1+5\varepsilon)$-approximation for $d(G_{in},G_{out},\mathcal{G})$ in the original graph $\mathcal{G}$, we get a $(1+5\varepsilon)\cdot(1+\varepsilon) \le (1+7\varepsilon)$-approximation of $d(G_{in},G_{in},\mathcal{G})$ in the original graph $\mathcal{G}$, once we apply recursion to $d(G_{in},G_{in},G^+_{in})$.

We note that our recursion halts once $|G^+_{in}| \le (256\log n/\varepsilon)^4$ in which case we naively compute $d(G_{in},G_{in},G^+_{in})$ using APSP in time $O(|G^+_{in}|^2)$. Recall that even at this final point, the distances between marked vertices still obey the invariant.

2.4 Running time

We now examine the total running time of our algorithm. Let n denote the number of vertices in our original graph $\mathcal{G}$ and let $V(G)$ denote the vertex set of the graph G in the current invocation of the recursive algorithm. The current invocation approximates $d(G_{in},G_{out},G_t)$ as shown in subsection 2.2 in time $O(|V(G)|/\varepsilon + 2^{O(1/\varepsilon)})$. It then goes on to construct the subgraphs G^+_{in} and G^+_{out} as shown in subsection 2.3, where we have that after contraction using dense portals, $|V(G^+_{in})| = \alpha|V(G)| + O(\log^4 n/\varepsilon^4)$ and $|V(G^+_{out})| = \beta|V(G)| + O(\log^4 n/\varepsilon^4)$, where $\alpha,\beta \le 2/3$ and $\alpha+\beta \le 1$. The time to construct $|V(G^+_{in})|$ and $|V(G^+_{out})|$ is dominated by the time required to compute SSSP for each dense portal, which requires $O(|V(G)|\cdot\log n/\varepsilon)$. We then continue recursively to G^+_{in} and

to G^+_{out}. Hence, if $T(|V(G)|)$ denotes the running time for G then we get that

$$\begin{aligned} T(|V(G)|) = O\big(|V(G))| \cdot \log n/\varepsilon + 2^{O(1/\varepsilon)}\big) \\ + T\big(\alpha|V(G)| + O(\log^4 n/\varepsilon^4)\big) \\ + T\big(\beta|V(G)| + O(\log^4 n/\varepsilon^4)\big). \end{aligned}$$

In the recursion's halting condition, once we get to components of size $|V(G)| = (256 \log n/\varepsilon)^4$, we naively run APSP. This takes $O(|V(G)|^2)$ time for each such component, and there are $O(n/|V(G)|)$ such components, so the total time is $O(n \cdot |V(G)|) = O(n \log^4 n/\varepsilon^4)$. It follows that

$$T(n) = O(n \log^4 n/\varepsilon^4 + n \cdot 2^{O(1/\varepsilon)}).$$

3 Concluding Remarks

We presented the first $(1+\varepsilon)$-factor approximation algorithm for the diameter of an undirected planar graph with non-negative edge lengths. Moreover, it is the first algorithm that provides a nontrivial (i.e. less than 2-factor) approximation in near-linear time.

It might still be possible to slightly improve the running time of our algorithm by removing a logarithmic factor, or by replacing the exponential dependency on ε with a polynomial one. In addition, the technique of Abraham and Gavoille [1] which generalizes shortest-path separators to the class of H-minor free graphs may also turn out to be useful.

References

1. I. Abraham and C. Gavoille. Object location using path separators. In *Proceedings of the 25th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 188–197, 2006.
2. D. Aingworth, C. Chekuri, P. Indyk, and R. Motwani. Fast estimation of diameter and shortest paths (without matrix multiplication). *SIAM Journal on Computing*, 28(4):1167–1181, 1999.
3. P. Berman and S.P Kasiviswanathan. Faster approximation of distances in graphs. In *Proceedings of the 10th International Workshop on Algorithms and Data Structures (WADS)*, pages 541–552, 2007.
4. K. Boitmanis, K. Freivalds, P. Ledins, and R. Opmanis. Fast and simple approximation of the diameter and radius of a graph. In *Proceedings of the 5th International Workshop on Experimental Algorithms (WEA)*, pages 98–108, 2006.
5. T.M. Chan. More algorithms for all-pairs shortest paths in weighted graphs. In *Proceedings of the 39th ACM Symposium on Theory of Computing (STOC)*, pages 590–598, 2007.
6. F. R. K. Chung. Diameters of graphs: Old problems and new results. *Congressus Numerantium*, 60:295–317, 1987.
7. F.F. Dragan, F. Nicolai, and A. Brandstadt. LexBFS-orderings and powers of graphs. In *Proceedings of International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, pages 166–180, 1996.
8. D. Eppstein. Subgraph isomorphism in planar graphs and related problems. In *Proceedings of the 6th Annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 632–640, 1995.
9. A.M. Farley and A. Proskurowski. Computation of the center and diameter of outerplanar graphs. *Discrete Applied Mathematics*, 2:185–191, 1980.
10. G. N. Frederickson. Fast algorithms for shortest paths in planar graphs. *SIAM Journal on Computing*, 16:1004–1022, 1987.
11. M. R. Henzinger, P. N. Klein, S. Rao, and S. Subramanian. Faster shortest-path algorithms for planar graphs. *Journal of Computer and System Sciences*, 55(1):3–23, 1997.
12. D.B. Johnson. Efficient algorithms for shortest paths in sparse graphs. *Journal of the ACM*, 24:1–13, 1977.
13. P. N. Klein. Multiple-source shortest paths in planar graphs. In *Proceedings of the 16th Annual ACM-SIAM Symposium On Discrete Mathematics (SODA)*, pages 146–155, 2005.

14. P.N. Klein. Preprocessing an undirected planar network to enable fast approximate distance queries. In *Proceedings of the 13th Annual ACM-SIAM Symposium On Discrete Mathematics (SODA)*, pages 820–827, 2002.
15. R.J Lipton and R.E. Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Math*, 36:177–189, 1979.
16. K. Mulmuley, U. Vazirani, and V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7:105–113, 1987.
17. S. Olariu. A simple linear-time algorithm for computing the center of an interval graph. *International Journal of Computer Mathematics*, 34:121–128, 1990.
18. Seth Pettie. A faster all-pairs shortest path algorithm for real-weighted sparse graphs. In *Proceedings of the 29th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 85–97, 2002.
19. M. Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *Journal of the ACM*, 51(6):993–1024, 2004. Announced at FOCS 2001.
20. C. Wulff-Nilsen. Wiener index, diameter, and stretch factor of a weighted planar graph in subquadratic time. Technical report, University of Copenhagen, 2008.